An Innocent User's Perspective on the CAS

Robert Lupton

Princeton University

FNAL, 1st March 2004

Outline

Outline

I must declare an interest: I have been closely involved with Alex and Jim in working on the skyserver schema. But I'm not recusing myself from giving this talk. Or at least writing it.

Outline

Introduction

Outline

Introduction

The CAS's Strengths

Outline

Introduction

The CAS's Strengths

The `casJobs` Server

Outline

Introduction

The CAS's Strengths

The `casJobs` Server

Problems with the CAS

Outline

Introduction

The CAS's Strengths

The `casJobs` Server

Problems with the CAS

Alternatives; or, Is the CAS a Good Idea?

Outline

Introduction

Introduction

This presentation is intended as a discussion of the CAS from a user's point of view; it is *not* intended to contrast CAS with other astronomical databases, built upon other commercial databases.

Introduction

This presentation is intended as a discussion of the CAS from a user's point of view; it is *not* intended to contrast CAS with other astronomical databases, built upon other commercial databases.

We are only using a subset of the tools shipped with `SQL-Server`, (e.g. *Microsoft Query Analyser*); I have no experience of these tools, and as they are not available to random astronomical consumers, they are not really relevant to CAS. It is possible that we shall have to reconsider the decision not to give users access to these tools.

We have become blasé about the capabilities of `C` (and e.g. `FORTRAN` optimisers (e.g. optimal register allocation; constant subexpression elimination), but there are still limits. For example, the latest version of gcc doesn't properly optimise potentially aliased pointers, despite the use of the C99 keyword `restrict`.

We have become blasé about the capabilities of `C` (and e.g. `FORTRAN` optimisers (e.g. optimal register allocation; constant subexpression elimination), but there are still limits. For example, the latest version of gcc doesn't properly optimise potentially aliased pointers, despite the use of the C99 keyword `restrict`.

The state of the art in database optimisers seems to be many years behind that in programming languages such as `C`. It is important not to overestimate their capabilities.

The CAS's Strengths

The CAS's Strengths

N.b. I have almost no experience of using the 'official' CAS web interface for queries; almost all of my work with the CAS has been done using `emacs` *to communicate directly with the defined* `.asp` *interface.*

The CAS's Strengths

- The relational model maps well onto the sort of queries that most astronomers are likely to want to make; only simple joins are required to e.g. associate seeing with a list of objects, or spectra with their `photoObj`s.[1]

[1] This join is actually pre-computed in the `spectroPhotoObj` (sp?) table.

The CAS's Strengths

- The relational model maps well onto the sort of queries that most astronomers are likely to want to make; only simple joins are required to e.g. associate seeing with a list of objects, or spectra with their `photoObj`s.[1]
- This good fit extends to `SQL`'s `where` clauses, e.g.
  `where psfMag_g - psfMag_r < 1.5 and psfMag_i < 18`.

[1] This join is actually pre-computed in the `spectroPhotoObj` (sp?) table.

The CAS's Strengths

- The relational model maps well onto the sort of queries that most astronomers are likely to want to make; only simple joins are required to e.g. associate seeing with a list of objects, or spectra with their `photoObj`s.[1]
- This good fit extends to `SQL`'s `where` clauses, e.g. `where psfMag_g - psfMag_r < 1.5 and psfMag_i < 18`.
- The documentation is generally sufficient, at least for the basic tables. I have not tried to sort out e.g. the tiling information from the provided documentation.

[1] This join is actually pre-computed in the `spectroPhotoObj` (sp?) table.

The CAS's Strengths

- The relational model maps well onto the sort of queries that most astronomers are likely to want to make; only simple joins are required to e.g. associate seeing with a list of objects, or spectra with their `photoObj`s.[1]
- This good fit extends to `SQL`'s `where` clauses, e.g. `where psfMag_g - psfMag_r < 1.5 and psfMag_i < 18`.
- The documentation is generally sufficient, at least for the basic tables. I have not tried to sort out e.g. the tiling information from the provided documentation.
- CAS provides a good model for remote access to the SDSS data, especially for those who do not wish to maintain a local copy.

[1] This join is actually pre-computed in the `spectroPhotoObj` (sp?) table.

The casJobs Server

A very significant enhancement over the intial CAS is provided by the casJobs server, which allows a user to submit arbitrary transact-SQL and to create private databases.

The casJobs Server

A very significant enhancement over the intial CAS is provided by the casJobs server, which allows a user to submit arbitrary transact-SQL and to create private databases.

The casJobs server has been under development, and hasn't been stable enough for real work. This should be fixed in the recently-announced DR2 casJobs server.

The casJobs Server

A very significant enhancement over the intial CAS is provided by the casJobs server, which allows a user to submit arbitrary transact-SQL and to create private databases.

The casJobs server has been under development, and hasn't been stable enough for real work. This should be fixed in the recently-announced DR2 casJobs server.

Using casJobs's Transact-SQL features takes us further from an ANSI-standard database.

The casJobs Server

A very significant enhancement over the intial CAS is provided by the casJobs server, which allows a user to submit arbitrary transact-SQL and to create private databases.

The casJobs server has been under development, and hasn't been stable enough for real work. This should be fixed in the recently-announced DR2 casJobs server.

Using casJobs's Transact-SQL features takes us further from an ANSI-standard database.

Using temporary tables to store expensive result sets is a *very* nice feature.

The casJobs Server

A very significant enhancement over the intial CAS is provided by the casJobs server, which allows a user to submit arbitrary transact-SQL and to create private databases.

The casJobs server has been under development, and hasn't been stable enough for real work. This should be fixed in the recently-announced DR2 casJobs server.

Using casJobs's Transact-SQL features takes us further from an ANSI-standard database.

Using temporary tables to store expensive result sets is a *very* nice feature.

The ability to create private tables (of e.g. the RC3) is also very nice. This is added functionality, rather than a way of improving upon the original CAS.

Problems with the CAS

Problems with the CAS

We *still* don't have all of the data that has been processed at FNAL in the CAS! What is more, I believe that it is CAS loading concerns that prevented us from preprocessing a handful of runs with the very latest `photo`.

Problems with the CAS

We *still* don't have all of the data that has been processed at FNAL in the CAS! What is more, I believe that it is CAS loading concerns that prevented us from preprocessing a handful of runs with the very latest `photo`.

This is not intrinsic to the CAS, but it is a problem that we need to resolve with help from FNAL and JHU.

Problems with the CAS

We *still* don't have all of the data that has been processed at FNAL in the CAS! What is more, I believe that it is CAS loading concerns that prevented us from preprocessing a handful of runs with the very latest `photo`.

This is not intrinsic to the CAS, but it is a problem that we need to resolve with help from FNAL and JHU.

Some innocent queries take a *very* long time to run. These are often due to known bugs in the `SQL-Server` query optimiser (the 'book-mark bug'). Most of these bugs are fixed in the next release.

The range of available statistical functions is very limited; e.g. medians and clipped means are hard to achieve without complicated nested queries. These are fixed in the next release, but at the cost of departing from ANSI-standard SQL.

The range of available statistical functions is very limited; e.g. medians and clipped means are hard to achieve without complicated nested queries. These are fixed in the next release, but at the cost of departing from ANSI-standard `SQL`.

Some `group by` queries take a very long time to finish. This is despite the fact that operators such as `avg` take only a *single* pass to evaluate their values. I haven't got chapter-and-verse, but I suspect that these queries are taking significantly longer than a simple sweep of the entire database.

The range of available statistical functions is very limited; e.g. medians and clipped means are hard to achieve without complicated nested queries. These are fixed in the next release, but at the cost of departing from ANSI-standard `SQL`.

Some `group by` queries take a very long time to finish. This is despite the fact that operators such as `avg` take only a *single* pass to evaluate their values. I haven't got chapter-and-verse, but I suspect that these queries are taking significantly longer than a simple sweep of the entire database.

It is likely that these queries could be sped up by a combination of temporary tables, and advice from someone smarter and more experienced in `SQL`.

The range of available statistical functions is very limited; e.g. medians and clipped means are hard to achieve without complicated nested queries. These are fixed in the next release, but at the cost of departing from ANSI-standard SQL.

Some `group by` queries take a very long time to finish. This is despite the fact that operators such as `avg` take only a *single* pass to evaluate their values. I haven't got chapter-and-verse, but I suspect that these queries are taking significantly longer than a simple sweep of the entire database.

It is likely that these queries could be sped up by a combination of temporary tables, and advice from someone smarter and more experienced in SQL.

It is also possible that these queries could be sped up by carefully tuning the database hardware.

It's a nuisance not being able to index tables (e.g. `psfMag_u` rather than `psfMag[0]`). I cannot write a query,and then run it again for a different band, and I cannot write a catalogued procedure[2] and pass in an index.

[2]Well, that's what they are called in JCL

It's a nuisance not being able to index tables (e.g. `psfMag_u` rather than `psfMag[0]`). I cannot write a query,and then run it again for a different band, and I cannot write a catalogued procedure[2] and pass in an index.

Some of this is circumvented by my `skyserver` mode for `emacs` that provides a macro processor that looks a little like `transact-SQL`.

[2] Well, that's what they are called in JCL

- It's a nuisance not being able to index tables (e.g. `psfMag_u` rather than `psfMag[0]`). I cannot write a query,and then run it again for a different band, and I cannot write a catalogued procedure[2] and pass in an index.

  Some of this is circumvented by my `skyserver` mode for `emacs` that provides a macro processor that looks a little like `transact-SQL`.

- There is no support for symbolic names (e.g. for the `SATUR` flag, or the value of `dbo.fSpecClass('qso')`). This is remedied by `casJobs` and by `emacs`'s `skyserver`.

[2] Well, that's what they are called in JCL

Alternatives; or, Is the CAS a Good Idea?

Alternatives; or, Is the CAS a Good Idea?

I am not convinced that using the CAS is faster than writing some special-purpose code to achieve my desires.

Alternatives; or, Is the CAS a Good Idea?

I am not convinced that using the CAS is faster than writing some special-purpose code to achieve my desires.

For statistical operations (which would be done by a `group by` in SQL, I think that writing a little piece of `C` (or `IDL` or even (maybe) `SM`) and reading the `FITS` tables would probably be faster than using `SQL` — and this allows for the time that it would take to write (and maybe compile) the code.

Alternatives; or, Is the CAS a Good Idea?

I am not convinced that using the CAS is faster than writing some special-purpose code to achieve my desires.

For statistical operations (which would be done by a `group by` in SQL, I think that writing a little piece of `C` (or `IDL` or even (maybe) `SM`) and reading the `FITS` tables would probably be faster than using `SQL` — and this allows for the time that it would take to write (and maybe compile) the code.

It is not clear to me that this would still be true if the DB were a thousand times bigger; it is possible that the database management system would do a much better job of distributing queries over many machines.

The same holds for point queries (on e.g. magnitudes and colours); if I'm serious about querying the entire database with a not-very restrictive set of cuts, I suspect that there's not (yet?) much gain from using the CAS.

The same holds for point queries (on e.g. magnitudes and colours); if I'm serious about querying the entire database with a not-very restrictive set of cuts, I suspect that there's not (yet?) much gain from using the CAS.

For 2-point queries ('find me pairs of objects that satisfy...') the CAS is very convenient. Of course, I could create some auxiliary `FITS` tables to make this easier — but now I'm inventing my own specialised database.

The same holds for point queries (on e.g. magnitudes and colours); if I'm serious about querying the entire database with a not-very restrictive set of cuts, I suspect that there's not (yet?) much gain from using the CAS.

For 2-point queries ('find me pairs of objects that satisfy...') the CAS is very convenient. Of course, I could create some auxiliary `FITS` tables to make this easier — but now I'm inventing my own specialised database.

This comparison with a local `C` query is rather unfair, as in the latter case I have total control over my local hardware configuration, and this isn't true of the CAS.

Summary

Summary

The syntax and functionality of the CAS (especially as augmented as `casJobs`) is exactly what we need.

Summary

The syntax and functionality of the CAS (especially as augmented as `casJobs`) is exactly what we need.

The speed with which some queries complete is in practice frustratingly slow.

Summary

The syntax and functionality of the CAS (especially as augmented as `casJobs`) is exactly what we need.

The speed with which some queries complete is in practice frustratingly slow.

A combination of more experience, more smarts, and more careful tuning, may go a long way to resolving these problems.